MATHEMATICAL MODELING AND COMPUTING, Vol. 10, No. 3, pp. 733–747 (2023)

Important subgraph discovery using non-dominance criterion

Ouaderhman T., Chamlal H., Oubaouzine A.

Department of Mathematics and Informatics,
Fundamental and Applied Mathematics Laboratory,
Faculty of Sciences Ain Chock, Hassan II University,
Casablanca, Morocco

(Received 18 February 2023; Accepted 17 July 2023)

Graph mining techniques have received a lot of attention to discover important subgraphs based on certain criteria. These techniques have become increasingly important due to the growing number of applications that rely on graph-based data. Some examples are: (i) microarray data analysis in bioinformatics, (ii) transportation network analysis, (iii) social network analysis. In this study, we propose a graph decomposition algorithm using the non-dominance criterion to identify important subgraphs based on two characteristics: edge connectivity and diameter. The proposed method uses a multi-objective optimization approach to maximize the edge connectivity and minimize the diameter. In a similar vein, identifying communities within a network can improve our comprehension of the network's characteristics and properties. Therefore, the detection of community structures in networks has been extensively studied. As a result, in this paper an innovative community detection method is presented based on our approach. The performance of the proposed technique is examined on both real-life and synthetically generated data sets.

Keywords: *graph mining; non-dominated subgraph; connectivity.*
2010 MSC: 05C69, 05C85, 91D30, 68T20 **DOI:** 10.23939/mmc2023.03.733

1. Introduction

Data mining is a significant discipline that aims to extract meaningful and valuable information from vast amounts of data. This field involves various steps such as data cleaning and preparation, exploration, visualization, model building, and evaluation. Data can be represented in multiple formats, including text, images, and other forms. Among the diverse forms of data representation, graphical representation stands out as a fascinating approach. For instance, bioinformatics and social networks are some examples of domains where graphical representation of data is commonly used.

Graph mining is a widely studied area of data mining that involves the discovery of hidden and valuable information in graphs. It has gained a lot of attention due to its wide range of applications, including social network analysis, bioinformatics, image processing, internet networks and road networks. Different approaches have been used to analyze graph-based data, which are generally based on clustering or subgraph mining [1]. Graph clustering is a technique that groups the vertices of a given input graph into clusters or groups based on certain similarities. Subgraph mining, on the other hand, involves the discovery of subgraphs based on an objective function [2]. Subgraph mining is considered one of the most challenging tasks in graph mining, as it involves discovering meaningful subgraphs based on a criterion specific to each author's goal.

A graph, denoted G, is composed of two sets: vertices, represented by V_G, and edges, represented by E_G. Moreover, each vertex of the graph has a label. Relational graphs, in particular, are distinguished by their unique labels. This paper focuses on the analysis of relational graphs.

Multi-objective optimization is a critical area of research, primarily because real-world problems often involve multiple objectives. This technique is crucial to support multi-criteria decision making, as it identifies the solution that best matches the decision maker's preferences from a set of satisfactory trade-off solutions. Unlike single-objective optimization, multi-objective optimization problems require the simultaneous optimization of several objectives, which often conflict with each other. Therefore,

© 2023 Lviv Polytechnic National University

there is no single optimal solution, and the concept of a set of Pareto optimal solutions emerges. Real-world optimization problems often have several conflicting objectives, and the optimal solutions are a set of points representing the best possible trade-offs between these objectives. The Pareto front, also known as the set of efficient solutions or the trade-off surface, represents an equilibrium, as improvement in one objective will result in degradation of at least one other objective. To determine the best solutions, we use the dominance relation as a criterion in our paper. In our case, a solution represents a subgraph S, which is a set of nodes and edges.

Preference-based processing is a technique used in database systems to return a specified number (k) of points with the highest ranking scores, or a set of points that are not dominated by any other points, based on their values [3]. This can be achieved through either top-k processing or skyline processing. In top-k processing, the k points with the highest scores are returned to the user, while in skyline processing, the returned points are those that are not dominated by any other points [4]. Consider two points, denoted i and j, each point consisting of k attributes. We say that point i dominates point j in the p-th attribute if and only if the p-th attribute value of i is greater than the corresponding value of j, denoted $i_p > j_p$. The skyline processing, as described by [4], works without relying on the use of a ranking function or a predetermined threshold.

To discover important subgraphs, the edge connectivity and the diameter of each subgraph are recorded as two attributes or objectives. The efficiency of a discovered subgraph increases as the edge connectivity increases and the diameter decreases. The best subgraphs are those that are maximized in terms of edge connectivity and minimized in terms of diameter.

The main idea of this paper is to propose a non dominated subgraph detection algorithm, which repeats the process of decomposing a graph into two smaller subgraphs by finding a minimal cut of the initial graph.

The proposed method discovers all subgraphs that are not dominated according to the two following objectives: the edge connectivity and the diameter. We aim to employ our methodology to detect communities of an input graph G.

The rest of the paper is structured as follows. Section 1 discusses the related work in the area of graph mining. Section 2 constitutes the bulk of this paper. It discusses in great detail the proposed method. Experimental results based on real-life as well as random generated data sets are offered in Section 3 and Section 4 in the form of applications in the field of social network analysis and community detection.

2. Related work

In this section, we present some fundamental contributions related to graph mining methods.

Cluster analysis is a crucial task in scientific research, which aims to identify groups of observations with high similarity within each group and low similarity between different groups. This technique has a variety of applications in a range of fields, including biology, medicine, and economics, among others. Edge connectivity is a well known concept in graph theory and it has been used as a measure of subgraph importance.

In [5], the edge connectivity has been applied as a clustering tool to group vertices with high levels of connectivity into distinct clusters.

Recent research has focused on multi-objective subgraph mining. An algorithm called Skygraph, introduced by Papadopoulos et al. [4], focuses on two specific objectives, namely the order and the edge connectivity. The main objective is to identify the optimal Pareto subgraphs that represent the non dominated subgraphs. The approach uses non-dominance relation to maximize two objectives, namely the order and the edge connectivity. In a similar vein, a multi-objective problem has been addressed by Prakash et al. [6]. They proposed an approach for subgraph mining that aims to maximize both support and subgraph size simultaneously. This technique identifies a set of non dominated subgraphs for both objectives. The authors later introduced another graph objective, the subgraph diameter, to address a three-objective subgraph exploration problem [7].

Exploring the dense regions of a graph is one of the effective techniques employed in graph mining. The density of a graph refers to the proportion of edges present in a graph G compared to the maximum number of edges it can contain. When a graph is complete, whether it is directed or not, its density is equal to 1, which means that it is entirely dense and cannot contain any additional edges. Maximilien Danisch et al. have proposed an efficient algorithm to compute locally dense exact decompositions in real-world graphs with multiple edges [8]. The algorithm uses the Frank–Wolfe algorithm, which is similar to gradient descent. It would be interesting to evaluate this algorithm in hypergraphs.

Quite recently, considerable attention has been paid to one of the most used methods called the K-core decomposition. The K-core decomposition consists of finding the largest subgraph of a network, in which each vertex has at least K neighbors in the subgraph. This method is based on a pruning process that consists in recursively removing vertices whose degree is lower than K [9]. In [10], the paper introduces a new approach for graph decomposition between K-core and graph density. It is a new tool for analyzing graphs.

In the study of complex networks, there is a growing interest in community detection. This topic has been widely explored in the field of network analysis, with a large body of literature devoted to the development of algorithms and methods for identifying the underlying community structure in a network.

Since it is not realistic to evaluate all previously proposed algorithms for community detection, this section will highlight a selection of notable algorithms.

The Girvan–Newman algorithm is an important community detection method in network science. Its objective is to divide a network into clusters by progressively eliminating edges based on their betweenness centrality, which measures how often an edge is located along the shortest paths between nodes in the network. First, the algorithm calculates the betweenness centrality of all edges in the network, and then eliminates the edge with the highest value. The algorithm then recalculates the betweenness centrality for the remaining edges and again eliminates the edge with the highest value. This process continues until there are no more edges in the network. The resulting clusters correspond to the connected components that emerge as edges are removed. The Girvan–Newman algorithm has been successfully used in many real-world networks, including social, biological, and transportation networks, due to its familiarity and ease of use. However, it has some limitations that should be considered. First, it can be very computationally intensive, as the algorithm involves computing the betweenness centrality for all edges at each iteration, which can be time consuming for large networks. Second, the algorithm is sensitive to the order in which edges are removed, which can result in different groupings depending on the starting conditions. Finally, the algorithm can produce unexpected results, such as splitting a large cluster into several smaller clusters or combining two separate clusters into a single cluster. To address these limitations, researchers have devised various modifications and extensions of the Girvan–Newman algorithm. One approach is to use different measures of centrality, such as degree centrality or eigenvector centrality, instead of betweenness centrality. Another approach is to include additional criteria for edge elimination, such as edge weight or node degree. A third approach is to use stochastic methods, such as Markov Chain Monte Carlo (MCMC), to randomly remove edges from the network and generate multiple solutions. These modifications and extensions have improved the accuracy and reliability of the Girvan–Newman algorithm in different types of networks.

Another popular approach is the Louvain method which is a popular approach for detecting communities in networks. It is known for its speed and optimization based technique for partitioning networks into communities. The method involves iteratively identifying nodes with the highest modularity gain and subsequently moving them to clusters that improve the overall clustering quality of the network. The final result is the most optimal community structure based on the chosen modularity function. However, the Louvain method is not without limitations. For example, it only works with non overlapping communities and relies on an a priori choice of modularity function that may not accurately reflect the community structure of the network. To overcome these limitations, the researchers devel-

oped modifications such as taking overlapping communities into account, using advanced optimization algorithms and additional constraints.

In addition, spectral clustering is a graph-based approach to identify communities within a network using linear algebra. It consists of transforming the network data into a lower-dimensional representation in which the communities are identifiable. This is accomplished through the construction of a graph Laplacian matrix, which represents the network structure. The eigenvectors of the matrix are then used for clustering in the lower dimensional space, typically using hierarchical clustering or k-means algorithms. Spectral clustering is well suited for handling complex network structures, including overlapping communities and hierarchical relationships between communities. In addition, it is computationally efficient, making it suitable for large-scale network analysis. Despite its advantages, spectral clustering has several limitations. In particular, it is sensitive to the pre-specified number of clusters, the similarity measure used in the construction of the graph's Laplacian matrix, and the lack of probabilistic interpretation of community assignments, which makes it difficult to assess the uncertainty of the results.

For more complete information on community detection methods, they can be found in [11, 12].

However, to the authors' best knowledge, very few publications are available in the literature that discuss the issue of partitioning a network to detect its communities. In a recent study [13], a new method for community detection was presented, which consists of dividing a network into two smaller communities by using a minimum cut. The betweenness centrality metric was used to calculate the value of the minimum cut, as this metric provides a measure of the centrality of a vertex within a graph. A key limitation of this method is that it assumes that the number of communities is known in advance, which make finding the optimal number of communities a challenging problem.

We go beyond this area of subgraph mining, by proposing a technique to discover subgraphs that are significant with respect to certain graph mining objectives such as the edge connectivity and the diameter. In this way, only the non dominated subgraphs are retained. Moreover, we introduce an innovative community detection method based on our approach.

3. Important subgraph discovery

The main objective of the proposed algorithm is to discover the most important and significant subgraphs of a given graph G.

The discovery of an important subgraph is determined by two objectives, the first one is the edge connectivity denoted λ_G and the second one is the diameter denoted $\mathrm{diam}(G)$.

In this section, we start by giving the most important definitions for our problem: the edge connectivity, the diameter and the non-dominance criterion. Then, we describe in details the algorithm.

Definition 1. *The edge connectivity, λ_G, of a connected graph is the minimum number of edges that must be removed in order to decompose the graph into two connected subgraphs.*

Definition 2. *The diameter,* $\mathrm{diam}(G)$*, of a graph G is the longest distance between any two vertices in the graph. It is calculated as the maximum number of edges that must be traversed to go from one vertex to another.*

Our goal is to determine the most significant subgraphs in terms of edge connectivity and diameter.

Each subgraph g is represented as a pair $(\lambda_g, \mathrm{diam}(g))$, where λ_g is the edge connectivity and $\mathrm{diam}(g)$ is the diameter.

The multi-objective subgraph mining problem treated in this contribution is defined considering a general definition of multi-objective optimization problems. In our case, a solution is a subgraph S, a set of nodes and edges.

The subgraph S is evaluated considering d different objectives on the subgraph's characteristics, such as the edge connectivity and the diameter.

Given a set of graphs G, we seek to extract the optimal Pareto–subgraphs of G defined by two objectives:

$$F(S) = \big(f_1(S), f_2(S)\big), \tag{1}$$

where $f_1 = \max \lambda_{G_i},\ i = 1, \ldots, n,\ f_2 = \min \operatorname{diam}(G_i),\ i = 1, \ldots, n,\ n$ is the number of subgraphs produced during the decomposition. For more details, see the next subsection of the algorithm.

Solution to the problem in Eq. (1) is a set of optimal subgraphs. To make comparison between any pair of subgraphs, we apply the well known criteria of dominance.

Non-dominance criterion involve comparing different solutions based on the trade-offs between multiple objectives.

In the context of graphs, the objectives may represent different characteristics or properties of the subgraphs, such as their size, connectivity, or centrality. In our case, the objectives are the edge connectivity and the diameter.

Between two subgraphs u and v, u dominates v if one of the following holds:
(1) $\lambda_u > \lambda_v$ and $\operatorname{diam}(u) < \operatorname{diam}(v)$;
(2) $\lambda_u = \lambda_v$ and $\operatorname{diam}(u) < \operatorname{diam}(v)$;
(3) $\lambda_u > \lambda_v$ and $\operatorname{diam}(u) = \operatorname{diam}(v)$.

3.1. The algorithm

The sum of the edges across different subgraphs is defined as the *cut* of the graph. To find the best way of decomposing the graph G, the problem is equivalent to find the minimum value of the cut.

The algorithm is based on successive applications of a min cut algorithm [14]. A minimum cut or a min-cut of a graph G is a partition of the vertices of a graph into two disjoint subsets. The application of successive min-cuts has been also used in SkyGraph as a tool to discover important subgraphs (2008) [4].

The proposed algorithm aims to partition a connected initial graph G into two smaller subgraphs by finding the minimum cut [14] that separates them. This is done repeatedly until each subgraph has only three vertices. Once all of the subgraphs have been calculated, the non-dominance criterion is applied based on two objectives: edge connectivity and diameter.

The min cut algorithm is described in the next paragraph.

Min cut algorithm. The Min cut of a weighted graph is the smallest possible total weight of the edges that must be removed to partition the graph into two separate groups. [14] introduced an algorithm to find the minimum cut in an undirected weighted graph.

The Algorithm 1 represents the min cut algorithm for an undirected weighted graph.

In our case, we focus on unweighted graphs.

A vertex k is said to be most tightly connected to a vertex set A if the sum of the weights of the edges connecting k and A is the highest among the vertices not belonging to A.

Algorithm 1 Min cut algorithm.

```
1: Function: MinCutPhase(Graph G, Weights W, Vertex a):
2: A < −a
3: while (A ≠ V)
4:    add tightly connected vertex to A
5:    save the cutofthephase and reduce the size of the graph G by combining the last two added vertices.
6: minimum = INF
7: Function:  MinCut(Graph G, Weights W, Vertex a):
8: while (|V| > 1)
9:    MinCutPhase(G,W,a)
10:   if (cutofthephase < minimum) then
11:      minimum = cutofthephase
12: return minimum
```

The algorithm operates by repeatedly merging the most closely connected vertices until there is only one node left in the graph. At each step of this process, the weight of the merged cut is recorded in a list. The minimum cut value for the graph is then determined by finding the smallest value in this list.

Let $G = (V, E, w)$ be a weighted undirected graph.

In the MinCutPhase function, the vertex a is a fixed point that is not changed throughout the process. The set A is a group of vertices that starts with vertex a and gradually adds more vertices from the graph G until it includes all vertices in the set V which is the collection of all vertices. A tightly connected vertex to A is a vertex whose edge weight to any vertex in A is the highest.

In the MinCut function, the global minimum cut is recorded and updated after each MinCut-Phase. When this process is completed for $n-1$ phases, the minimum cut for the entire graph can be determined.

In the proposed algorithm, we are looking for maximizing the edge connectivity and minimizing the diameter. The best subgraphs are the ones that are not dominated by any another subgraph.

The proposed algorithm that gonna deal with edge connectivity and diameter is as follows:

–> Input: initial input graph G;

–> Output: $ND(G)$, set of the non dominated subgraphs of G.

The proposed algorithm can be divided into three steps:

- step of preprocessing;
- step of decomposition;
- step of non-dominance criterion.

The preprocessing phase. In this step, we apply two preprocessing methods to the graph G:

– The first one is isolated vertices detection. An isolated vertex is a vertex of a graph that has no edges, a vertex with degree zero.

– The second one is bridge detection. A bridge is an edge of a graph whose removal increases the number of connected components of the graph.

The algorithm of the first step is given in the Algorithm 2.

Algorithm 2 The first step: Preprocessing phase

Given a graph G
if (G has one or more isolated vertices) **then**
    remove isolated vertices of G
if (G has one or more bridges) **then**
    remove bridges of G
return The graph G

The decomposition phase. After detecting the isolated vertices, the bridges and removing them, we introduce the decomposition phase.

Algorithm 3 The second step: Decomposition phase

n_G: number of vertex of a graph G
λ_G: edge connectivity of a graph G
diam(G): the diameter of a graph G
run min-cut algorithm on G
create subgraphs G_{left} and G_{right}
compute number of vertex, connectivity and diameter for G_{left} and G_{right}: ($n_{G_{left}}$, $\lambda_{G_{left}}$, diam(G_{left})) ; ($n_{G_{right}}$, $\lambda_{G_{right}}$, diam(G_{right}))
while ($n_{G_{left}} > 3$ and $n_{G_{right}} > 3$)
    run min-cut algorithm on G_{left} and G_{right}
    create G_{left} and G_{right} again for each two subgraph produced
    compute number of vertex, connectivity and diameter for each two subgraph produced
return A =(set of all subgraphs produced during the decomposition)

The algorithm is based on successive applications of the min cut algorithm.

A cut is a set of edges whose removal disconnects the graph. The min cut algorithm is an algorithm that computes the minimum edges that removal them decompose the graph into two subgraphs: G_{left} and G_{right}.

We apply the min cut algorithm until each subgraph consists of three vertex: the stopping criterion of the algorithm.

The algorithm of the second step is given in the Algorithm 3.

Non-dominance criterion phase. The second step of decomposition returns a set of all the subgraphs that have been produced during the decomposition.

We apply the non-dominance criterion to those subgraphs according to the two following objectives: the edge connectivity λ_G and the diameter $\text{diam}(G)$.

We are looking for maximizing the edge connectivity and minimizing the diameter.

For solving such a multi-objective optimization problem, the dominance relation is required.

The algorithm of the third step is given in the Algorithm 4.

Algorithm 4 The third step: Non dominance criterion phase

1: A: set of all the subgraphs produced during the decomposition
2: Apply the non-dominance criterion to the set of subgraphs A
3: **return** $ND(G)$: set of the non dominated subgraphs

The proposed full algorithm with the three steps is given in the Algorithm 5.

Algorithm 5 Graph decomposition with non-dominance criterion

1: n_G: number of vertex of a graph G
2: λ_G: edge connectivity of a graph G
3: $diam(G)$: the diameter of a graph G
4: **if** (G has one or more isolated vertices) **then**
5: remove isolated vertices of G
6: **if** (G has one or more bridges) **then**
7: remove bridges of G
8: run min-cut algorithm on G
9: create subgraphs G_{left} and G_{right}
10: compute number of vertex, connectivity and diameter for G_{left} and G_{right}: ($n_{G_{left}}$, $\lambda_{G_{left}}$, $\text{diam}(G_{left})$); ($n_{G_{right}}$, $\lambda_{G_{right}}$, $\text{diam}(G_{right})$)
11: **while** ($n_{G_{left}} > 3$ and $n_{G_{right}} > 3$)
12: run min-cut algorithm on G_{left} and G_{right}
13: create G_{left} and G_{right} again for each two subgraph produced
14: compute number of vertex, connectivity and diameter for each two subgraph produced
15: compute $ND(G)$ for all the subgraphs produced during the decomposition.
16: **return** $ND(G)$

3.2. Example

We present an example (Figure 1) for our proposed method, focusing on the last two steps of the algorithm: the graph decomposition and the non-dominance criterion.

Given a connected undirected graph $G = (1, 2, 3, 4, 5, 6, 7)$ which is composed of 7 vertices and 10 edges.

The edge connectivity and the diameter is computed for each subgraph produced during the decomposition.

The initial graph G has edge connectivity $\lambda_G = 1$. The removal of the edge between the vertex 2 and the vertex 4 produce two subgraphs:

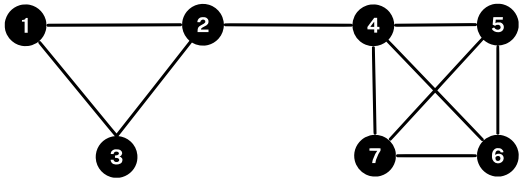

Fig. 1. Graph G.

–> The first subgraph: $g_1 = (1, 2, 3)$ with $\lambda_{g_1} = 2$ and $\text{diam}(g_1) = 1$.

–> The second subgraph: $g_2 = (4, 5, 6, 7)$ with $\lambda_{g_2} = 3$ and $\text{diam}(g_2) = 1$.

The stopping criterion (number of vertex > 3) is not satisfied for the subgraph g_2, then the algorithm is applied again and it produces two subgraphs:

–> The first subgraph: $g_3 = (4, 5, 6)$ with $\lambda_G = 2$ and $\text{diam}(g_3) = 1$.

–> The second subgraph: $g_4 = (7)$ with $\lambda_G = 0$ and $\text{diam}(g_4) = 0$.

By applying the criterion of non-dominance for all the subgraphs (g_1, g_2, g_3, g_4) produced during the decomposition we realize that the non dominated subgraph is the graph g_2 which has a high edge

connectivity, a minimal diameter and who is not dominated by any another subgraph produced in the decomposition of the graph G.

The decomposition of the example is given in Figure 2. It represents a tree-decomposition of the graph G.

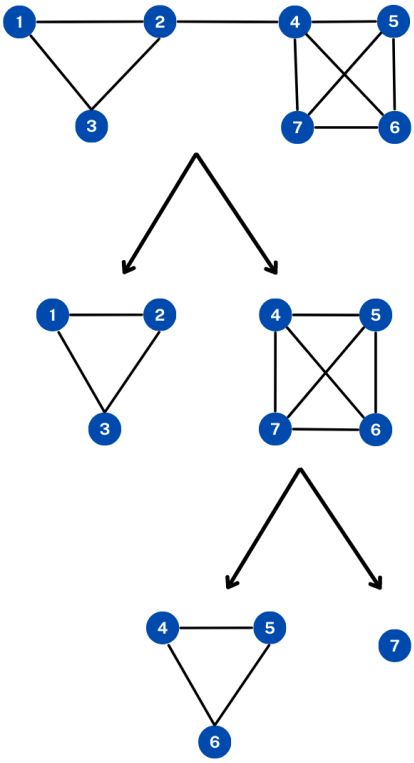

Fig. 2. The decomposition of the graph G.

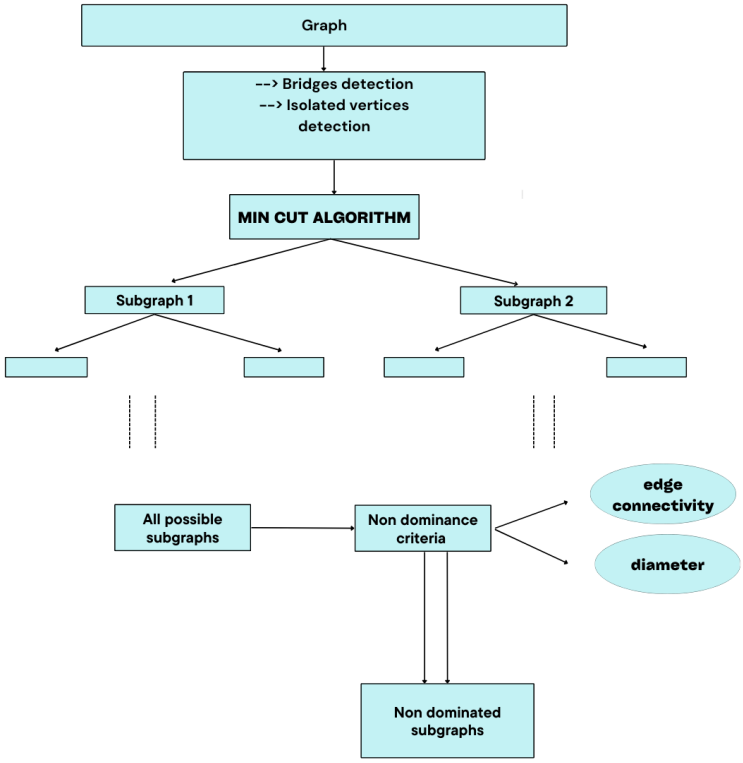

Fig. 3. Flowchart of the proposed method.

3.3. The complexity of the algorithm

Let V and E denote the number of vertices and the number of edges of the graph.

To remove an isolated vertex, we need to find the vertex in the graph. The time complexity of this operation is $\mathrm{O}(V)$.

A fast bridge detection algorithm based on Depth First Search (DFS) is applied before implementing the minimum cut process. The running time of this algorithm is linear with respect to the number of vertices (V) and edges (E) in the graph, meaning it takes $\mathrm{O}(V+E)$ time to execute.

The proposed method is based on the min cut algorithm. [15] have given a bound of $O(V \cdot E + V^2 \cdot \log V)$ for undirected graphs (weighted and unweighted graphs).

In our case, we focus on unweighted graphs and therefore the min cut can be computed in $\mathrm{O}(V \cdot E)$.

Figure 3 illustrates the proposed method with the 3 steps of the algorithm.

4. Application 1: Social network analysis

Networks are a common feature in our world, including both physical networks such as road networks and virtual networks like social media. They can be useful for making informed decisions by analyzing them.

Each network consists of:

–> Vertices: the individuals for whom the network is being constructed.

–> Edges: represent the connections between vertices and signify the relationship between the individuals in the network.

The performance evaluation study is based on real life data sets and they are summarized in Table 1.

One of the most interesting and studied domain in graph mining is social network analysis. It deals with examining the relationships and interactions between individuals in a social network through the use of graph theory and statistical techniques. The aim of this analysis is to identify the fundamental structure of relationships and connections within the network and how they impact human behavior and decision-making. This domain has applications in various fields including psychology, sociology, computer science, and marketing. With the rise of online social networks, the significance of social network analysis has only increased, as it allows for the analysis of vast amounts of data generated by these networks. However, this also calls for the development of innovative and more advanced methods to extract meaningful information from this data.

For our proposed method, we have used several datasets from social networks, more precisely the social network Facebook.

Table 1. Data sets used in performance evaluation.

Network	Num vertices	Num edges	Min degree	Max degree	Average degree
Relationships network	39	94	1	12	4.82
Social network 1	351	982	1	347	5.59
Social network 2	536	1823	1	347	6.80
Social network 3	1397	2716	2	305	3.89
Social network 4	1506	3238	1	1045	4.30

The performance of the algorithm when we applied it to the real-life data sets is given in Table 2.

In addition to the total running time, the table contains information about some measurements applied to the algorithm, for example the number of bridges detected which is one of the techniques used in the preprocessing step of the algorithm.

Moreover, the cardinality of $ND(G)$ that represents the number of non dominated subgraphs determined for each data set and it is given in the last column of Table 2.

Table 2. Performance results for real-life data sets.

Network	Min cut computations	Bridges detected	Running time (sec)	$ND(G)$
Relationships network	34	2	1	1
Social network 1	245	103	50	14
Social network 2	305	228	120	4
Social network 3	307	1087	121	14
Social network 4	324	1179	173	12

Tables 3 and 4 present the results obtained for random graphs.

Each graph contains 100 vertices for Table 3, while the number of edges varies between 200 and 1000.

Table 4 contains for each graph 500 vertices while the number of edges varies between 500 and 2000.

Each column is associated with a different graph.

Table 3. Performance results for random graphs (we set number of vertices to 100 and number of edges varies between 200 and 1000).

Measurements	100/200	100/400	100/600	100/800	100/1000
Running time (sec)	5	5	6	7	7
Min cut computations	89	96	97	97	97
Bridges detected	4	1	0	0	0
$ND(G)$	3	3	6	8	16

The performance of the algorithm when we applied it to random graphs is given in Tables 3 and 4. For both real and random graphs, the performance of the algorithm depends strongly on the number of vertices and the number of edges.

The efficiency of the algorithm depends on the type of input graph.

Table 4. Performance results for random graphs (we set number of vertices to 500 and number of edges varies between 500 and 2000).

Measurements	500/500	500/1000	500/1500	500/2000
Running time (sec)	24	229	306	315
Min cut computations	173	446	486	490
Bridges detected	210	38	3	1
$ND(G)$	13	85	40	169

The bridge detection, that represents one of the tools in the step of preprocessing, is an interesting technique for reducing the number of min cut computation and the running time.

Table 5 contain a comparison of the running time before and after applying the bridge detection for real datasets.

We notice that there is a big difference between before and after applying the bridge detection.

For example, the running time for the social network 2 before applying the bridge detection is 456 seconds while after applying the technique, the algorithm took 120 seconds to do the decomposition and calculate the non dominated subgraphs. The same thing for other networks, the running time after applying the technique is too small when compared to before.

Table 5. Comparison of the running time before and after applying bridge detection for real datasets.

Network	Running time before bridge detection (sec)	Running time after bridge detection (sec)
Relationships network	2	1
Social network 1	149	50
Social network 2	456	120
Social network 3	1800	121
Social network 4	2100	173

Table 6 contains a comparison of the min cut computation before and after applying the bridge detection for real datasets.

For each network, the number of decomposition made by applying the bridge detection is smaller comparing before applying the technique.

For example, for the network 3, the algorithm made 1396 decompositions to reach the result if we do not apply the detection, while applying it, the number of decompositions made is 307 which is a very small number compared to 1396.

Table 6. Comparison of the min cut computation before and after applying bridge detection for real datasets.

Network	Min cut computation before bridge detection	Min cut computation after bridge detection
Relationships network	36	34
Social network 1	348	245
Social network 2	533	305
Social network 3	1396	307
Social network 4	1505	324

We can conclude that the bridge detection optimization is an efficient and an important tool which allows to minimize the number of decompositions and to minimize the running time.

5. Application 2: Detection of communities

In network analysis, identifying and defining communities is a central task in understanding the structure and organization of complex systems.

One approach to finding communities is to identify subgraphs that are dense in terms of the connections between nodes, and where the nodes within the subgraph are not easily connected to

nodes outside of the subgraph. In this context, non dominated subgraphs with high edge connectivity and minimal number of nodes can be considered as potential communities.

Edge connectivity measures the minimum number of edges that need to be removed from a graph to disconnect it. A high edge connectivity value indicates that the nodes within the subgraph are strongly connected to each other.

The minimal number of nodes in the subgraph also suggests that the subgraph is dense and has a high proportion of edges relative to the number of nodes.

Together, these characteristics suggest that the subgraph is a cohesive group or community of nodes that are tightly connected to each other and less connected to the rest of the graph.

In general, the min cut algorithm does not always give a balanced cut, it can reduce in each decomposition a single vertex.

Let $ND = (ND_1, \ldots, ND_k)$ be the set of non dominated subgraphs and v a single vertex which represents a subgraph produced during the decomposition.

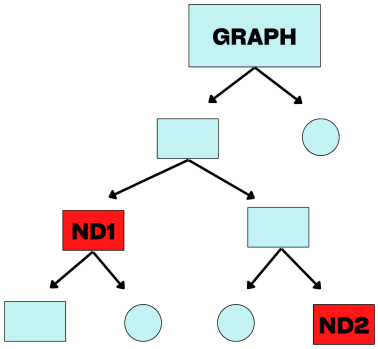

Fig. 4. Example of decomposition.

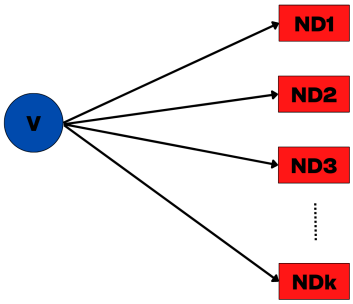

Fig. 5. Connectivity of v with the set ND.

The non dominated subgraphs are considered as potential communities but subgraphs that contain only a single vertex should also form a community to cover all the vertices of the graph and to have all the possible communities.

In this context, we propose a technique to solve this problem.

We introduce the following modularity technique. If there is a connectivity between each subgraph produced during the decomposition to the non dominated subgraphs (see Figure 5 which represents a single vertex connected with the set of non dominated subgraphs), we will use the modularity for forming communities and to cover all the vertices of the graph, otherwise they will belong to the parent subgraph.

The modularity technique is as follows:

–> For a vertex v:

$$Q^k(v) = \frac{1}{2m_k} \sum_{w \in ND_k} \left(A_{vw} - \frac{k_v \cdot k_w}{2m_k} \right),$$

where A_{vw} is the value of the adjacency matrix between vertices v and w, k_v is the sum of the weights of the edges adjacent to v, m_k is the number of edges of the ND_k.

–> For a subgraph of G with number of vertex >1:

$$Q^k(subG) = \frac{1}{2m_k} \sum_{u \in subG} \sum_{v \in ND_k} \left(A_{uv} - \frac{k_u \cdot k_v}{2m_k} \right).$$

For each vertex v or subgraph $subG$, we have to check if they have a connectivity with the non dominated subgraphs or not.

If there is a connectivity, we will compute their modularity for the non dominated subgraphs that are connected with, and the vertex or the subgraph with the highest modularity will be assigned to the non dominated subgraph with which it had maximum modularity. If there is not a connectivity, v or $subG$ will be added to their parent subgraph.

Figure 6 illustrates all the steps of the proposed approach to detect communities.

The first step is the preprocessing which contains two tools: the detection of isolated vertices and the detection of bridges. After detecting them, we delete them.

The second step is the graph decomposition step where we will apply the min cut algorithm until the stopping criterion: number of subgraphs vertices > 3.

After having computed all the possible subgraphs, we will apply in step 3, the non-dominance criterion to detect only the non dominated subgraphs according to the two following objectives: edge connectivity and diameter.

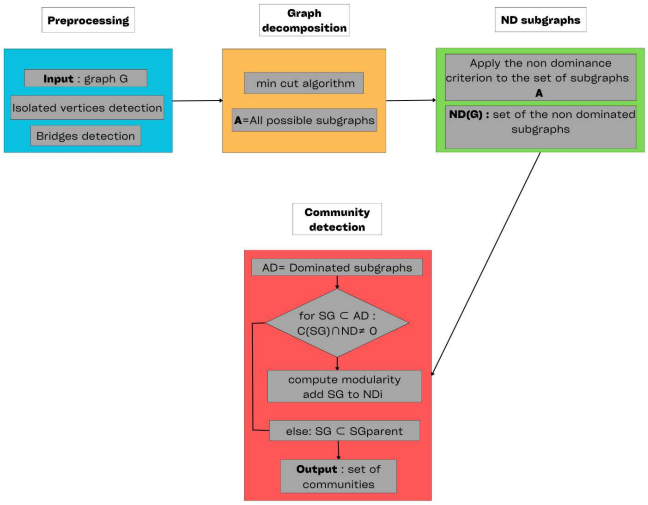

Fig. 6. Flowchart of the proposed approach to detect communities based on the non dominated subgraphs.

These non dominated subgraphs will be presented as kernels that we use in clustering to group similar groups.

The dominated subgraphs will be added to the non dominated subgraphs according to the connectivity and the modularity.

For each dominated subgraph, we need to check whether it has connectivity with the non dominated subgraphs or not(in the last step in Figure 6, $C(SG)$ means the connectivity of the subgraph SG).

If there is connectivity, we will compute the modularity for the non dominated subgraphs that are connected, and the dominated subgraph with the highest modularity will be assigned to the non dominated subgraph with which it had the maximum modularity.

If there is no connectivity, the dominated subgraph will be added to its parent subgraph.

Finally, in the output of the proposed approach, we detect the communities of the network.

Results. This section describes the results of applying the proposed approach to three example networks. The two first examples deals with simple networks. Those two examples checks the validity of the proposed technique.

After the simple validity check, the third example is used the show the performance of the proposed technique.

Example 1. Figure 7 shows the graph used in the example 1. After applying the Algorithm 5, the non dominated subgraphs are $ND_1 = (2, 3, 4)$ and $ND_2 = (5, 7, 8)$. For the subgraphs that just have one vertex are: $g_1 = (1)$, $g_2 = (6)$ and $g_3 = (9)$. For each single vertex, we calculate their connectivity with the non dominated subgraphs. The subgraph g_1 is connected with ND_1 while g_2 and g_3 are connected with ND_2. Because each vertex has a connectivity with a one non dominated graph, we will assign them directly without computing the modularity. The result of the technique used is shown in Figure 8. The example 1 has two obvious communities $C_1 = (1, 2, 3, 4)$ and $C_2 = (5, 6, 7, 8, 9)$.

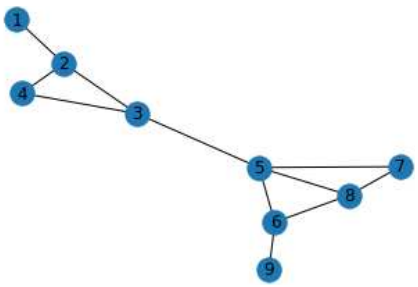

Fig. 7. Example 1.

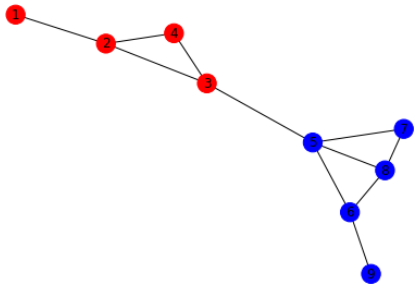

Fig. 8. Result.

Example 2. Figure 9 shows the graph used in the example 2. After applying the Algorithm 5, the non dominated subgraphs are $ND_1 = (2, 3, 4)$ and $ND_2 = (6, 8, 9)$. For the subgraphs that just have one vertex are: $g_1 = (1)$, $g_2 = (5)$, $g_3 = (9)$ and $g_4 = (10)$. For this example, we have the subgraph g_2 that contain a single vertex and which has a connectivity with the two non dominated subgraphs ND_1 and ND_2. We calculate its modularity to know for which non dominated subgraph we will assign it. For g_2 we have: $Q^1(g_2) = 0.177$, $Q^2(g_2) = 0.13$ and $Q^1(g_2) > Q^2(g_2)$. Then g_2 will be added to ND_1. For g_1 will be added to ND_1 while g_3 and g_4 will be added to ND_2 after calculating their connectivity. The result of the technique used is shown in Figure 10. The example 2 has two communities $C_1 = (1, 2, 3, 4, 5)$ and $C_2 = (6, 7, 8, 9, 10)$.

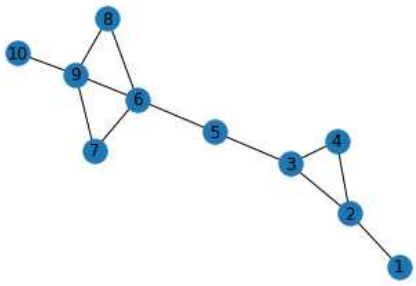

Fig. 9. Example 2.

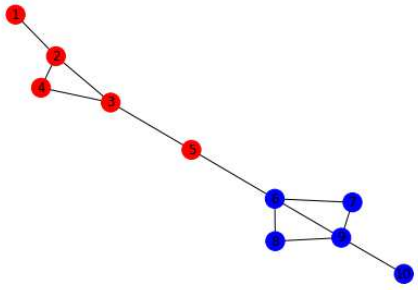

Fig. 10. Result.

Example 3. We use the real word network data: Zachary's karate club network. It is a network which represents social interactions between 34 individuals. After applying the Algorithm 5, the non dominated subgraph is $ND = (0, 1, 2, 3, 13)$. For the other subgraphs produced during the decomposition, we determine if they have a connectivity with the non dominated subgraph or not. If the connectivity exists, we add those subgraphs to the non dominated subgraph. If there is no connectivity, the subgraph will be incorporated into its parent subgraph. The result is shown in Figure 12. The example 3 has two communities $C_1 = (0, 1, 2, 3, 4, 5, 6, 7, 8, 10, 11, 12, 13, 16, 17, 19, 21)$ and $C_2 = (9, 14, 15, 18, 20, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33)$.

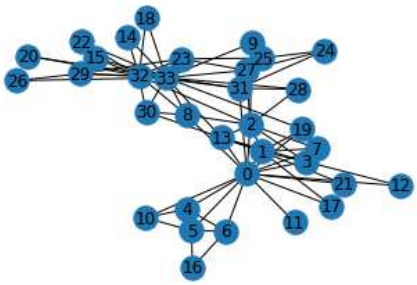

Fig. 11. Example 3.

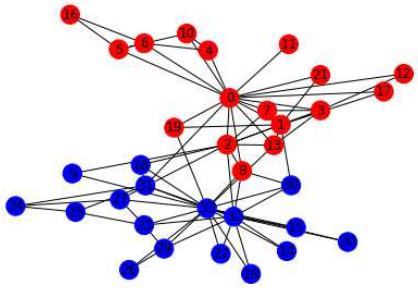

Fig. 12. Result.

6. Conclusion

In this paper, we have studied one of the most interesting problem of graph mining: the problem of discovering important subgraphs of an input graph G. The importance of the subgraph is based on edge connectivity and diameter. In order to achieve this goal, we have developed an algorithm to determine the non dominated subgraphs. This algorithm is based on successive min-cut decompositions of the initial graph G using two performance improvement mechanisms. The first one is the optimization *bridge detection* and the second one is the stopping criterion of the algorithm *number of vertex* $>$ 3 which are two tools for reducing the number of min-cut computations and avoiding unnecessary decompositions. We have notived the importance of the bridge detection tool following two measures: the running time and the number of decompositions.

For future work, we will focus on a criterion for setting the number of non dominated subgraphs (k) resulting from the algorithm. In this way, a user can control the number of subgraphs from the beginning by selecting a value for k. It would be interesting to add other optimizations that can be applied to speed up the discovery of non dominated subgraphs. These optimizations aim at reducing the number of min-cut computations.

Our approach have been used to detect communities within a graph. Further evaluation of the performance of the algorithm through the use of different benchmarks and larger network data is needed and remains a future task.

[1] Rehman S. U., Asmat U. K., Simon F. Graph mining: A survey of graph mining techniques. Seventh International Conference on Digital Information Management (ICDIM 2012). 88–92 (2012).

[2] Nandita B., Anmol R G. Graph-based data mining: A new approach for data analysis. International Journal of Scientific & Engineering Research. **5** (5), 1230–1236 (2014).

[3] Papadias D., Tao Y., Fu G., Seeger B. Progressive skyline computation in database systems. ACM Transactions on Database Systems (TODS). **30** (1), 41–82 (2005).

[4] Papadopoulos A. N., Lyritsis A., Manolopoulos Y. Skygraph: an algorithm for important subgraph discovery in relational graphs. Data Mining and Knowledge Discovery. **17**, 57–76 (2008).

[5] Hartuv E., Shamir R. A clustering algorithm based on graph connectivity. Information Processing Letters. **76** (4–6), 175–181 (2000).

[6] Shelokar P., Quirin A., Cordón Ó. A multiobjective evolutionary programming framework for graph-based data mining. Information Sciences. **237**, 118–136 (2013).

[7] Shelokar P., Quirin A., Cordón Ó. Three-objective subgraph mining using multiobjective evolutionary programming. Journal of Computer and System Sciences. **80** (1), 16–26 (2014).

[8] Danisch M., Chan T.-H. H., Sozio M. Large scale density-friendly graph decomposition via convex programming. WWW '17: Proceedings of the 26th International Conference on World Wide Web. 233–242 (2017).

[9] Kong Y.-X., Shi G.-Y., Wu R.-J., Zhang Y.-C. k-core: Theories and applications. Physics Reports. **832**, 1–32 (2019).

[10] Tatti N. Density-friendly graph decomposition. ACM Transactions on Knowledge Discovery from Data. **13** (5), 1–29 (2019).

[11] Lancichinetti A., Fortunato S. Community detection algorithms: A comparative analysis. Physical Review E. **80** (5), 056117 (2009).

[12] Cuvelier E., Aufaure M.-A. Graph Mining and Communities Detection. European Big Data Management and Analytics Summer School, eBISS 2011: Business Intelligence. 117–138 (2012).

[13] Shin H., Park J., Kang D. A graph-cut-based approach to community detection in networks. Applied Sciences. **12** (12), 6218 (2022).

[14] Stoer M., Wagner F. A simple min-cut algorithm. Journal of the ACM. **44** (4), 585–591 (1997).

[15] Nagamochi H., Ibaraki T. Computing edge-connectivity in multigraphs and capacitated graphs. SIAM Journal on Discrete Mathematics. **5** (1), 54–66 (1992).

Виявлення важливого підграфа за допомогою критерію недомінування

Уадерман Т., Чамлал Х., Убаузін А.

Кафедра математики та інформатики,
лабораторія фундаментальної та прикладної математики,
факультет наук Айн Чок, Університет Хасана II,
Касабланка, Марокко

Методам аналізу графів приділяли багато уваги для виявлення важливих підграфів на основі певних критеріїв. Ці методи стають все більш важливими через зростання кількості програм, які використовують дані на основі графіків. Деякі приклади: (i) аналіз даних мікроматриць у біоінформатиці, (ii) аналіз транспортних мереж, (iii) аналіз соціальних мереж. У цьому дослідженні пропонується алгоритм декомпозиції графа з використанням критерію недомінування для визначення важливих підграфів на основі двох характеристик: реберної зв'язності та діаметра. Запропонований метод використовує підхід багатоцільової оптимізації для максимізації зв'язності ребер і мінімізації діаметра. Подібно ідентифікація спільнот у мережі може покращити наше розуміння характеристик і властивостей мережі. Як наслідок, було детально вивчено виявлення структур спільноти в мережах. У статті подано інноваційний метод виявлення спільноти, заснований на запропонованому підході. Ефективність запропонованого метода перевіряється як на реальних, так і на синтетично згенерованих наборах даних.

Keywords: *виявлення графів; недомінований підграф; підключення.*